This application is submitted in the name of Inventor Sameer Tannous, Assignor to Cisco Technology, Inc., a California Corporation.

SPECIFICATION

SYSTEM AND METHOD FOR TESTING NETWORK PROTOCOLS

FIELD OF THE INVENTION

[0001] This invention relates to the testing of network protocols. More particularly, this invention relates to modifying protocols in a protocol stack during the testing of the network protocol. Still more particularly, this invention relates to receiving command in interpreted code and modifying a protocol in the protocol stack in response to the command.

PRIOR ART

[0002] Network protocols are used by processing device to communicate over a network connecting the processing devices. For purposes of this discussion, a processing device is a router, computer or other device having a processing system for processing digital data. A network protocol is made of a protocol stack. A protocol stack is a group of separate software modules or protocols that interact to transmit data between processing devices over the network. Each software module operates in a certain manner defined by a state machine. Each software module also sends a message between the machines to transfer data.

[0003] New network protocols are constantly being developed having new protocols stack and older network protocols are constantly being modified by changing the protocols in the protocol stack of the network protocol. When a new protocol is developed or an existing protocol is modified, developers desire to test the new protocol or improvements. Typically, the developers are testing protocol compliance, correctness, scalability and performance of the protocol stacks in the new network protocols or in improved existing network protocols.

[0004] It is a problem for designers of network protocols to develop systems for testing the network protocols. In order to test the network protocols, designers typically must program a unique test for each network protocol. Whenever a protocol in the network protocol is modified, the test must also be modified to test the modified protocol. To solve this problem, designers have attempted to describe the protocols through template. The templates allow decoding of a network protocol with a router executing a Pagent image of the network protocol. However, attempts to use templates have heretofore been unsuccessful.

[0005] Some test tools have been created to test network protocols that allow a designer to replay messages from a recorded file or allow a designer to create and send new messages. However, these test tools require a designer have extensive knowledge of the source code for the network protocol. Furthermore, these test tools also usually limit a designer to modifying only one protocol in the protocol

stack. Another problem is that these test tools may require that the code be complied again in order to create new messages. Thus, a test must be started over when a modification begins.

[0006] For these reasons, there is a need in the art for a test tool that provides a better system for modifying messages in protocols in a protocol stack of a network protocol during testing.

SUMMARY OF THE INVENTION

[0007] The above and other problems with the prior art are solved and an advance in the art is made by the system for testing a network protocol in accordance with this invention. A first advantage of a system in accordance with an embodiment of this invention is that messages may be added, modified, and/or deleted from any protocol in a protocol stack of a network protocol. A second advantage of a test system in accordance with an embodiment of this invention is that changes may be made to a state machine of any protocol in a protocol stack of the network protocol. A third advantage an embodiment of a test system in accordance with this invention is that the modification can be made to a protocol in the protocol stack at run time without the need to re-compile and restart testing. The testing begins by executing communications between a plurality of devices using the network protocol being tested. A command to modify one of the protocols in a protocol stack of the network protocol is received. The modification

indicated in the command is then performed on the protocol in the protocol stack identified in the command.

[0008] Preferably, the command is received in compiled code of the network protocol. This allows the modification to be inserted into interpreter without compilation. Thus, the modifications may be made at run time.

[0009] In a preferred exemplary embodiment, the following processes are used to modify messages in a protocol in the protocol stack of a network protocol in the following manner. First, when a command is received, the process determines the protocol in the protocol stack that is to be modified responsive to receiving the command.

[0010] The process may then determine whether the command is adding a message to the identified protocol. The message is then added to the identified protocol.

[0011] If the process determines the command is to remove a message from identified protocol, the message is removed from the identified protocol. If the process determines that a message in the identified protocol is to be modified, the process removes the message from the identified protocol and a message including the modification is added to the identified protocol.

[0012] The process may also determine whether a command is to modify a state machine of the identified protocol. If the command is determined to be modifying the state machine of the identified protocol, the process modifies the state machine of the identified protocol as indicated by the command.

BRIEF DESCRIPTION OF THE DRAWINGS

The above described and other features and advantage of this invention are described in the following detailed description and the following drawings:

[0013] Figure 1 illustrating a processing device that provides a testing system in accordance with this invention;

[0014] Figure 2 illustrating a communication system being tested by a testing system in accordance with this invention;

[0015] Figure 3 illustrating a conceptual diagram of a protocol stack of software applications in a system being tested.

[0016] Figure 4 illustrating a conceptual diagram of a testing system in accordance with this invention;

[0017] Figure 5 illustrating a flow diagram of a process for modifying a protocol in a protocol stack of a network protocol by a test system in accordance with this invention; and

[0018] Figure 6 illustrating a flow diagram of a process for performing modifications in accordance with this invention.

DETAILED DESCRIPTION

[0019] This invention relates to a system and method for providing a test system for network protocols used for communication between processing devices that allows a tester to modify a protocol in a protocol stack of the network protocol while the test is being performed. This invention is described below in a manner that allows one skilled in the art to design and use a system in accordance with this invention. When possible, like reference numerals have been used in the accompanying figures to describe the same or like elements of this invention.

[0020] Figure 1 illustrates a processing system 100 that performs application to provide a test system in accordance with this invention. One skilled in the art will recognize that although one configuration of one processing device is shown that this invention may be implemented on more than one processing device connected over a network.

[0021] Figure 2 illustrates an exemplary embodiment of a processing system 200. One skilled in the art will recognize that each device connected to network 100 in Figure 1 includes a processing system 200. However, the exact configuration and devices connected to the processing system in each individual device in the network may vary depending upon the functions that the processing device performs.

[0022] Processing system 100 has a Central Processing Unit (CPU) 101. CPU 101 is a processor, microprocessor, or any combination of processors and/or microprocessors that execute instructions stored in memory to perform an application. CPU 101 is connected to a memory bus 103 and Input/Output (I/O) bus 104.

[0023] A non-volatile memory such as Read Only Memory (ROM) 111 is connected to CPU 101 via memory bus 103. ROM 111 stores instructions for initialization and other systems command of processing system 100. One skilled in the art will recognize that any memory that cannot be written to by CPU 101 may be used for the functions of ROM 111.

[0024] A volatile memory such as Random Access Memory (RAM) 112 is also connected to CPU 101 via memory bus 204. RAM 112 stores instructions for all processes being executed and data operated upon by the executed processes. One skilled in the art will recognize that other types of memories such DRAM and SRAM may also be used as a volatile memory and that memory caches and other memory devices (not shown) may be connected to memory bus 104.

[0025] Peripheral devices including, but not limited to, memory 121, display 122, I/O device 123, and network connection device 124 that are connected to CPU 201 via I/O bus 104. I/O bus 104 carries data between the device and CPU

101. Memory 121 is a device for storing data unto a media. Some examples of memory 121 include read/write compact discs (CDs), and magnetic disk drives. Display 122 is a monitor or display and associated drivers that convert data to a display. I/O device 123 is a keyboard, a pointing device or other device that may be used by a user to input data. Network device 124 is a modem or Ethernet "card" that connects processing system 100 to a network. One skilled in the art will recognize that exact configuration and devices connected to each processing system may vary depending upon the operations of the processing system.

[0026] This invention relates to a test system for testing a network protocol. A network protocol is a system that provides communications between processing devices in a network. Figure 2 illustrates a diagram of a system 200 in which a first processing device 201 and a second processing device 202 are connected by a network 205 to allow the processing device to transmit data to each other. Data is transmitted by the processing devices in packets. A network protocol is the messages transmitted between the processing devices to assure the delivery of packets. The network protocol is formed by a protocol stack. A protocol stack includes the various software packets transmitted between the processing system. Typically, the data from lower protocol in the stack is encapsulated by the packet of a higher protocol in the stack.

[0027] Figure 3 illustrates a conceptual diagram of an exemplary protocol stack of a network protocol the first layer of protocol stack 300 is RUDP 301. The second layer of the stack is a session manager 303. The third layer is MTP3. The fourth layer is ISUP. The fourth and lowest layer is UDP. These protocols work together to transmit packets over network.

[0028] In order to test a network protocol, test system must emulate devices communicating over a network using the protocol. A test application is also executed which monitors the messages transmitted between the emulated devices and execution of applications in a processing device. Figure 4 illustrates a test system 400 being executed by a processing system. One skilled in the art will recognize that test system 400 may be executed by a single processing system or multiple processing systems connected by a network. Test system 400 includes software executing emulations of first processing device 405 and a second processing device 410 transmitting packets over path 415 using the network protocol under test. Preferably, each protocol in the protocol stack is modeled by a data object. Each data object modeling a protocol is de-coupled from the objects modeling other protocol. The de-coupling allows modifications to the data structures of the data objects while the test is being performed. One skilled in the art will recognize that more than two processing devices may be emulated.

[0029] Test application 420 is also executed. Test application also receives packets transmitted over path 415 via path 425 to monitor the messages being passed between the processing devices during a communication session. Internal messages and internal data generated during the communication session in first processing device 405 are received by the test application 410 via path 430. Internal messages and internal data generated during the communication session in second processing device 410 are received by the test application 410 via path 435.

[0030] This invention allows a test application to add new messages, delete messages, modify messages, and modify a state machine of any protocol in a protocol stack of a network protocol during the executing of a test of the network protocol. Figure 5 illustrates a preferred exemplary embodiment of a test application in accordance with this invention. Process 500 begins in step 505 by executing the test. The test application and emulation of all devices are executed in a conventional manner. Preferably, the test models each protocol in the protocol stack as a separate data object. Each data object is de-coupled from the data objects modeling the other protocols. This allows the data structures of the data objects to be changed while the test is being performed.

[0031] In step 510, a display requesting a command to modify a protocol in a protocol stack in the network protocol being tested. The display may be a window,

kernal, or box displayed on a user screen requesting input of a command. In step 515, a command to modify a protocol in the protocol stack received. Preferably, the command is received in interpreted code to allow direct insertion into the protocol. Also, preferably, the command includes the protocol in the protocol stack to be modified, the type of change, and possibly the device that is to receive the change to the protocol.

[0032] If the command is not in interpreted code, the command is compiled into translated code in step 520. Preferably, the compilation is completed without having to recompile the code for the entire protocol. This allows the test to continue during insertion since the entire protocol does not have to be re-compiled.

[0033] In step 525, process 500 determines which protocol in the protocol stack is to be modified by the command. In step 530, process 500 determines which device running the protocol is to be modified. One skilled in the art will recognize that the command may modify the protocol in one, multiple, or all the processing devices being emulated.

[0034] In step 535, the modification is performed on the protocol in the determined protocol stack in all determined device. This may include adding a

message, removing a message, modifying a message, or modifying a state machine of the protocol in protocol stack. After step 535, process 500 ends.

[0035] Figure 6 illustrate process 600 for performing the step 535 of process 500. Process 600 begins in step 605 where process 600 determines whether the command is adding a new message to the protocol. If the command is adding a new message, the new message is added to the protocol in step 607 and process 600 ends.

[0036] If the command is determined not to be adding a new message, process 600 determines whether a message is being removed from the protocol by the command in step 610. If the command is removing a message, the message is removed from the protocol in step 612.

[0037] If the command is determined not to be removing a message, process 600 determines whether the command is modifying a message in step 615. If the command is removing a message, the message is removed in step 617. In step 618, a new command is generated from the old command and includes the modifications to the message and adds the message to the protocol.

[0038] If the command is not for modifying a message, process 600 determines whether the command is to modify a state machine of the protocol in step 620. If

the command is to modify the state machine, the state machine is modified in step 622. The modification may include adding a new state, removing a state, modifying a state, or changing the jumps from one state to another state. If the command is not to modify the state machine, process 600 returns an error in step 630 and process 600 ends.

[0039] The above is a description of exemplary embodiments of this invention. Those skilled in the art can and will design alternative embodiments that can and will design alternative embodiments that infringe this invention as set forth in the following claims either literally or through the Doctrine of Equivalents.